Dual-level Parallel Analysis of Harbor Wave Response Using MPI and OpenMP

Steve W. Bova, Clay P. Breshears, Christine Cuicchi, Zeki Demirbilek, and Henry A. Gabb

CEWES MSRC and WES Coastal and Hydraulics Laboratory

Project Goals

- Apply the latest HPC technology to coastal operations & planning
- Modify CGWAVE, an existing serial, production code
  - Increase model resolution
  - Improve simulation turnaround time
  - Very little source code alteration

Applications

- Military and civil works
- Forecasting tool of DoD
- Harbors resonate at natural frequencies
  - Evaluate placement of wave gauges for harbor monitoring
  - Determine where problem mooring and on/off loading conditions may occur
  - Select optimum sites for amphibious operations

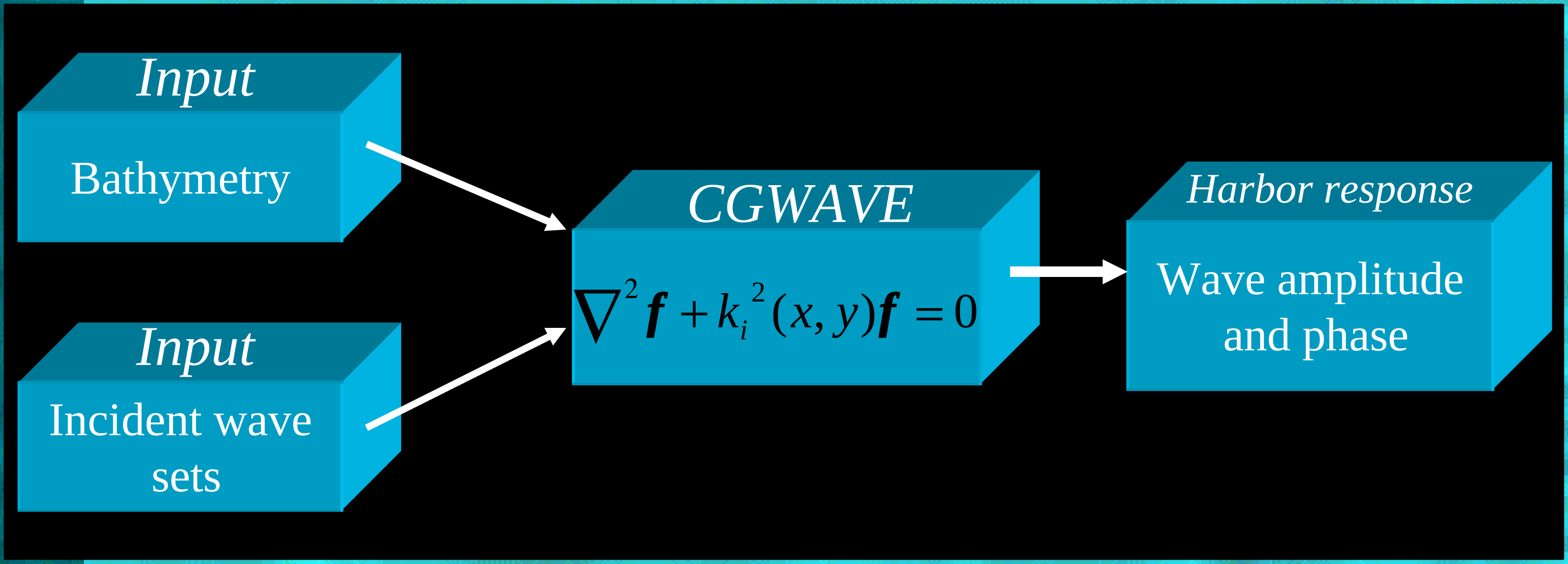
Computer model

- **CGWAVE**
  - **Serial code is production harbor wave climate and response tool**
  - **Results used by**
    - **US Department of Defense**
    - **Lloyds of London**
  - **Method**
    - **Elliptic mild-slope wave equation**
    - **Leads to an independent Helmholtz-type equation for each incident wave component**
    - **Resulting large, sparse systems solved via conjugate gradient**

Computer model

Two kinds of resolution

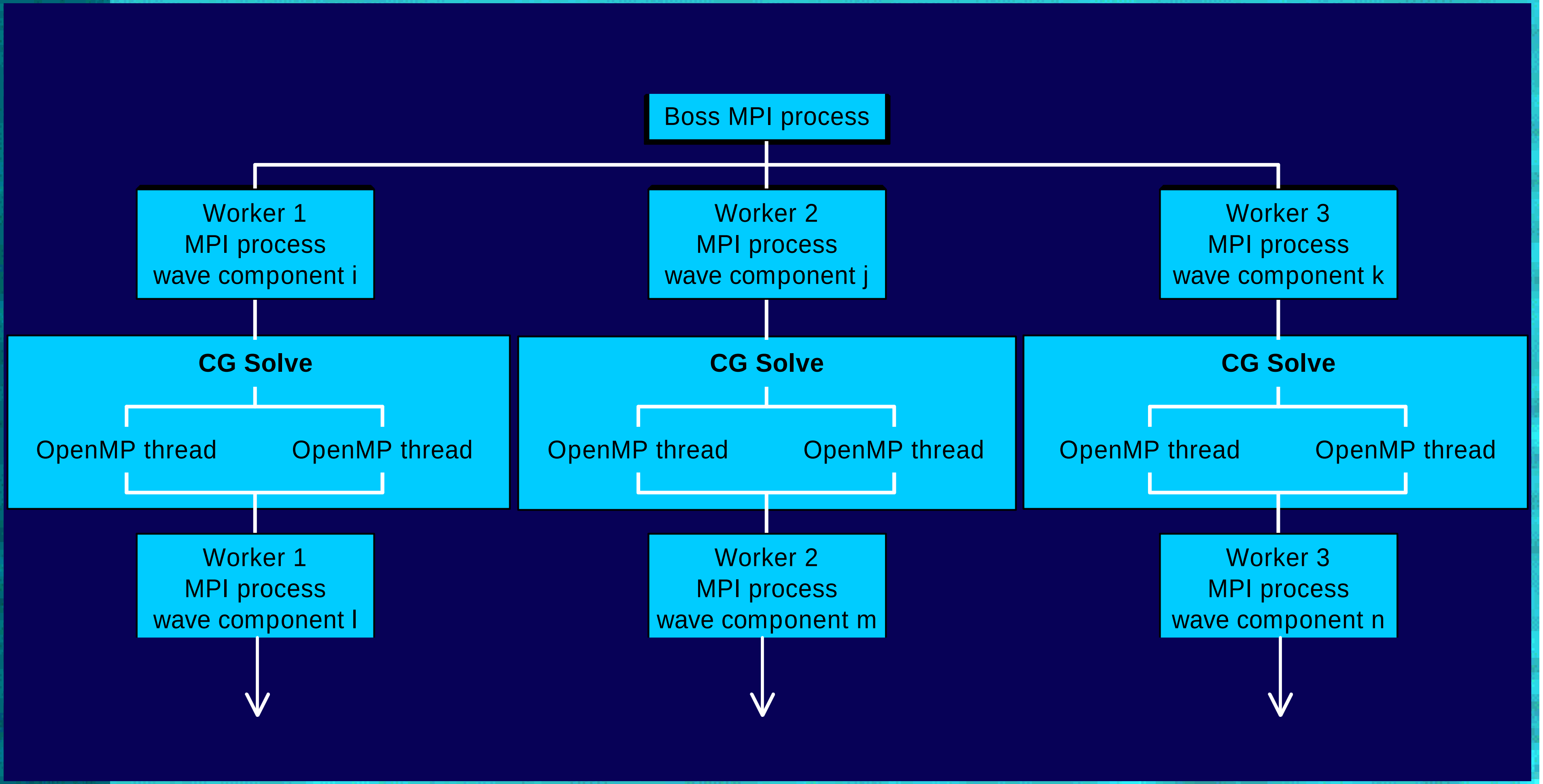
- spatial (finite element mesh, bathymetry)
- sea-state (number of incident waves)



Parallel implementation issues

- NUMA requires attention to data placement with OpenMP
  - “first touch” principle on SGI/CRAY Origin2000
- Two load-balancing schemes tested
  - Round-robin
    - static
    - efficiency depends on set ordering
  - Boss-worker
    - dynamic
    - independent of set ordering, system load

Dual-level parallelism



Dynamic load balancing

- **Boss:**

```
do i = 1, number_of_wave_components
    blocking receive  ! wait for work request
    blocking send     ! send work order
enddo
! All wave components solved
do worker = 1, nprocs - 1
    blocking receive  ! wait for work request
    blocking send     ! fire worker
enddo
MPI_Finalize
```

- **Worker:**

```
do infinite loop
    blocking send     ! ask boss for work
    blocking receive  ! get component
    if (not termination signal) then
          Perform calculations to solve wave
        component
    else
      exit infinite loop
    endif
enddo
MPI_Finalize
```

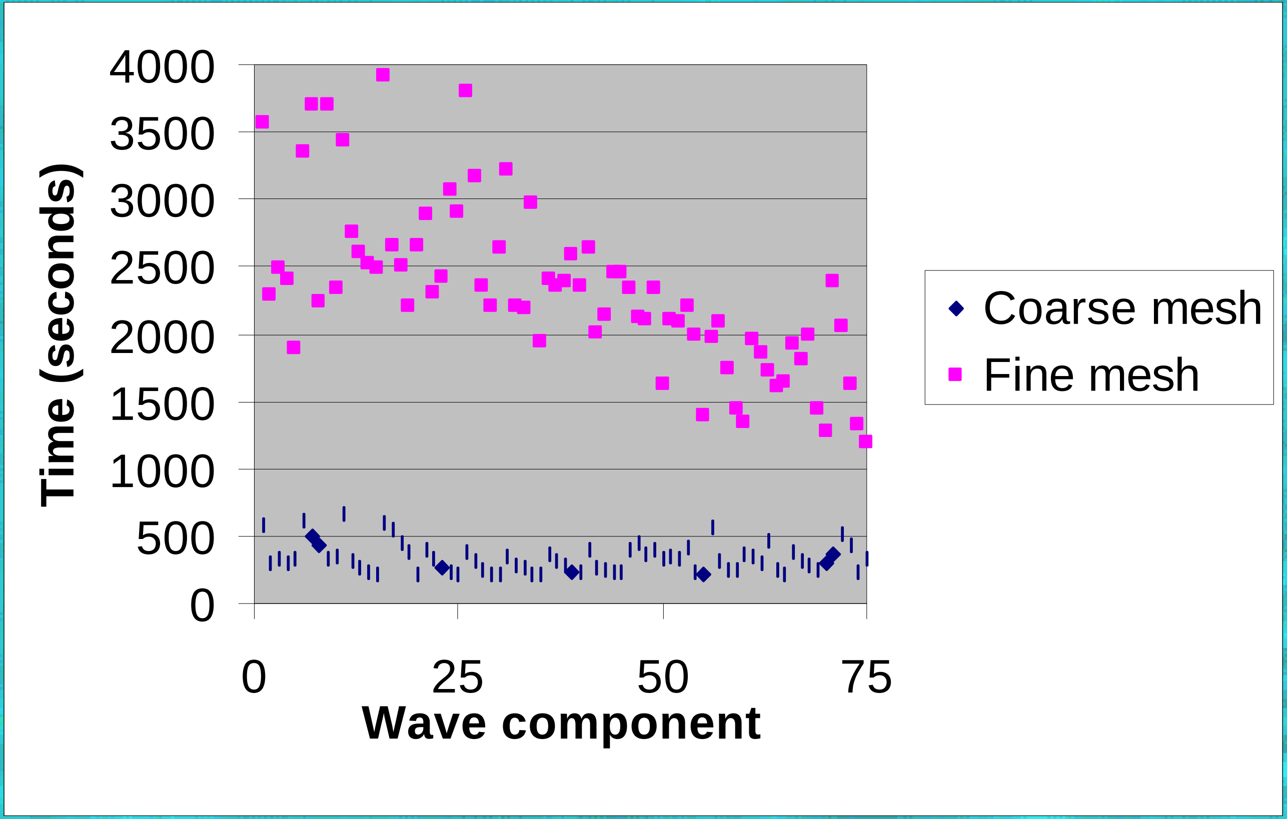
Two sample problem sets

- **Coarse mesh: 50,000 elements**
- **Fine mesh: 150,000 elements**

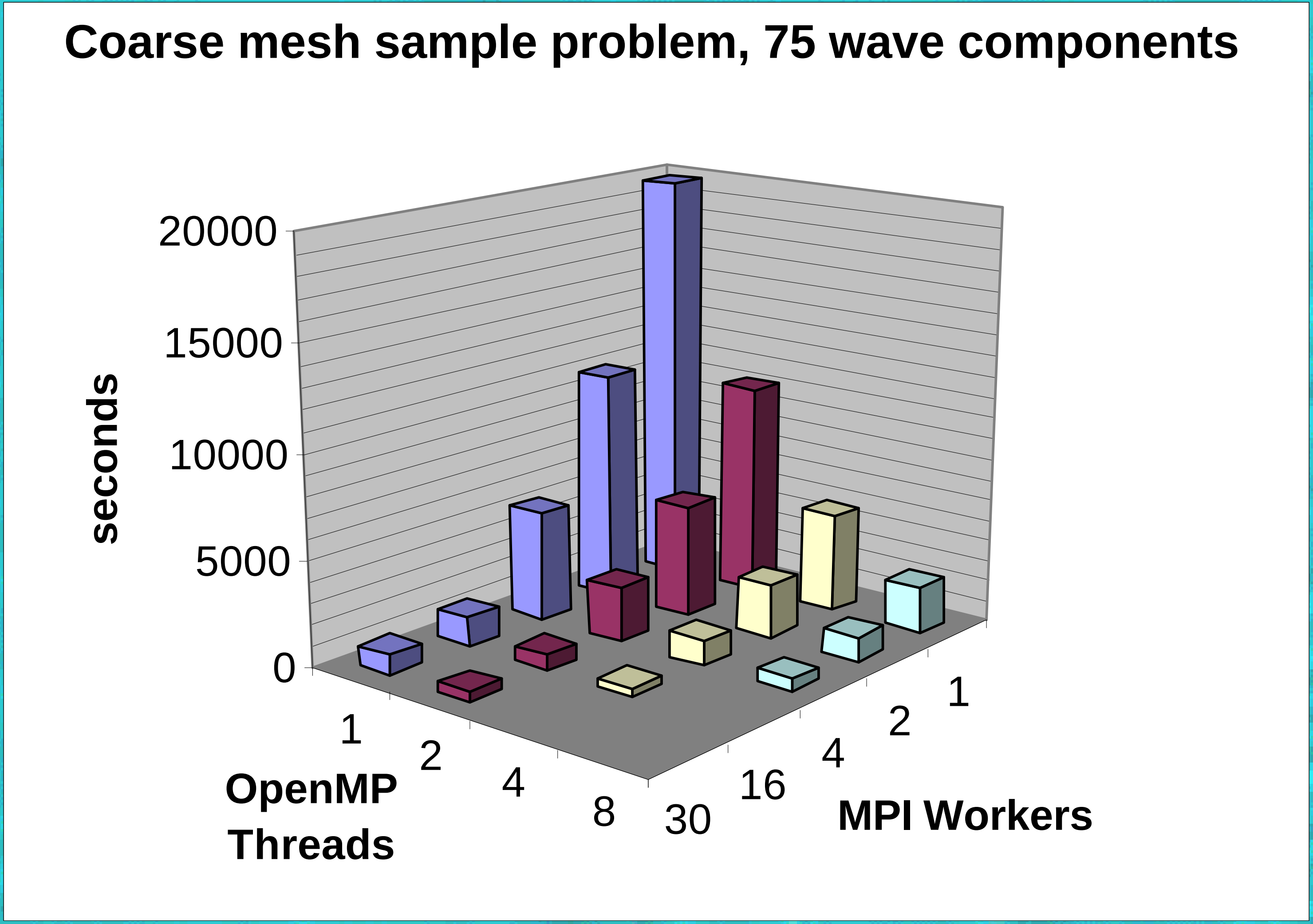
- **75 incident waves in sea state**
  - **five periods**
  - **15 directions**
  - **40 amplitudes**

Why load balancing is necessary

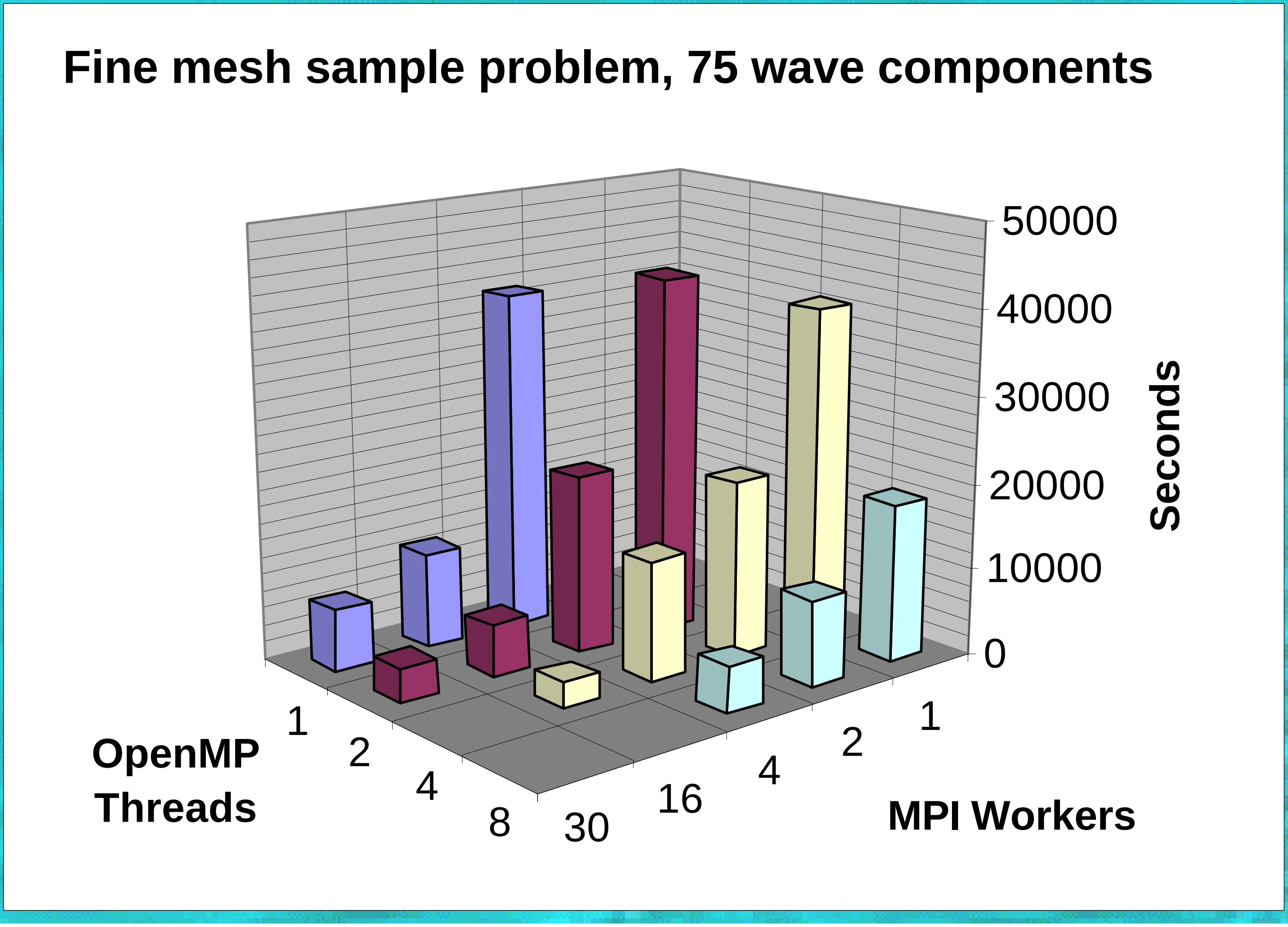
The chart shows the wallclock time required to solve each individual component



Load-balanced wallclock time

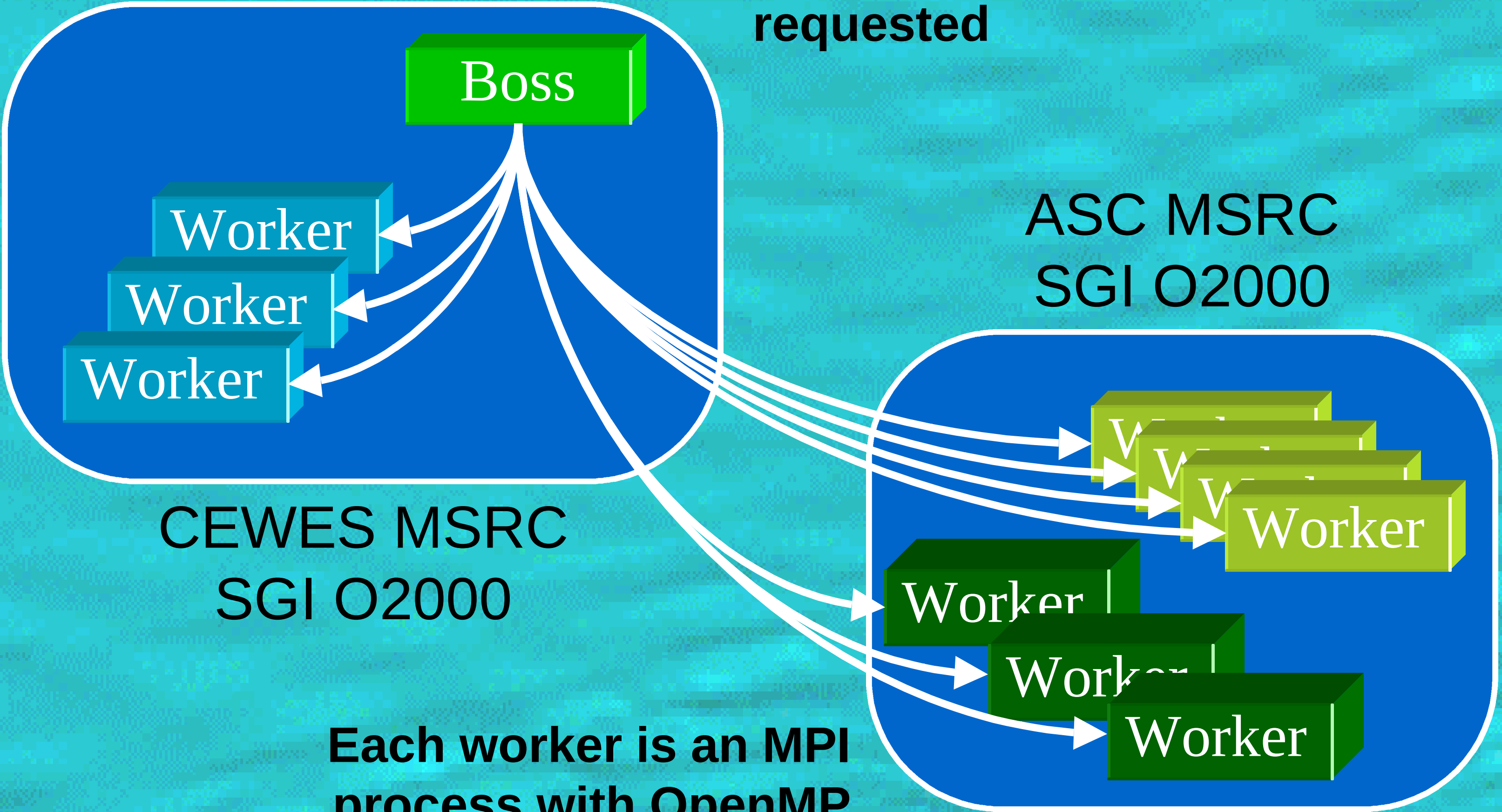


Load-balanced wallclock time



MPI_Connect

Boss process distributes components to workers as requested



Each worker is an MPI process with OpenMP threads

MPI_Connect algorithm

- **Boss:**

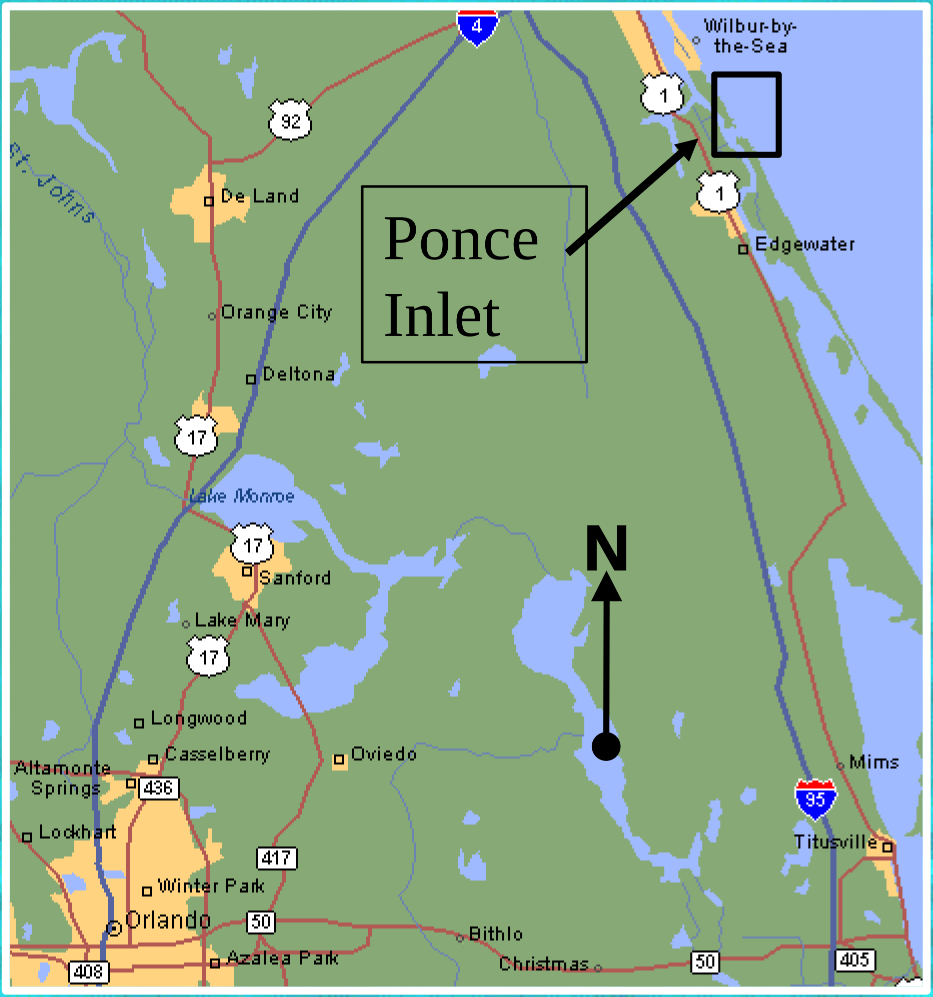
```
connect with worker groups  ! intercomm set-up
do i = 1, number_of_wave_components
    probe for worker request    ! busy wait on comm
    blocking receive
    blocking send     ! send work order
enddo
! All wave components solved
do worker = 1, nworkers
    probe for worker request    ! busy wait on comm
    blocking receive
    blocking send     ! fire worker
enddo
MPI_Finalize
```

- **Worker:**

```
connect with boss  ! intercomm set-up
do infinite loop
    blocking send      ! ask boss for work
    blocking receive   ! get component
    if (not termination signal) then
            Perform calculations to solve wave
            component
    else
      exit infinite loop
    endif
  enddo
  MPI_Finalize
```

Application

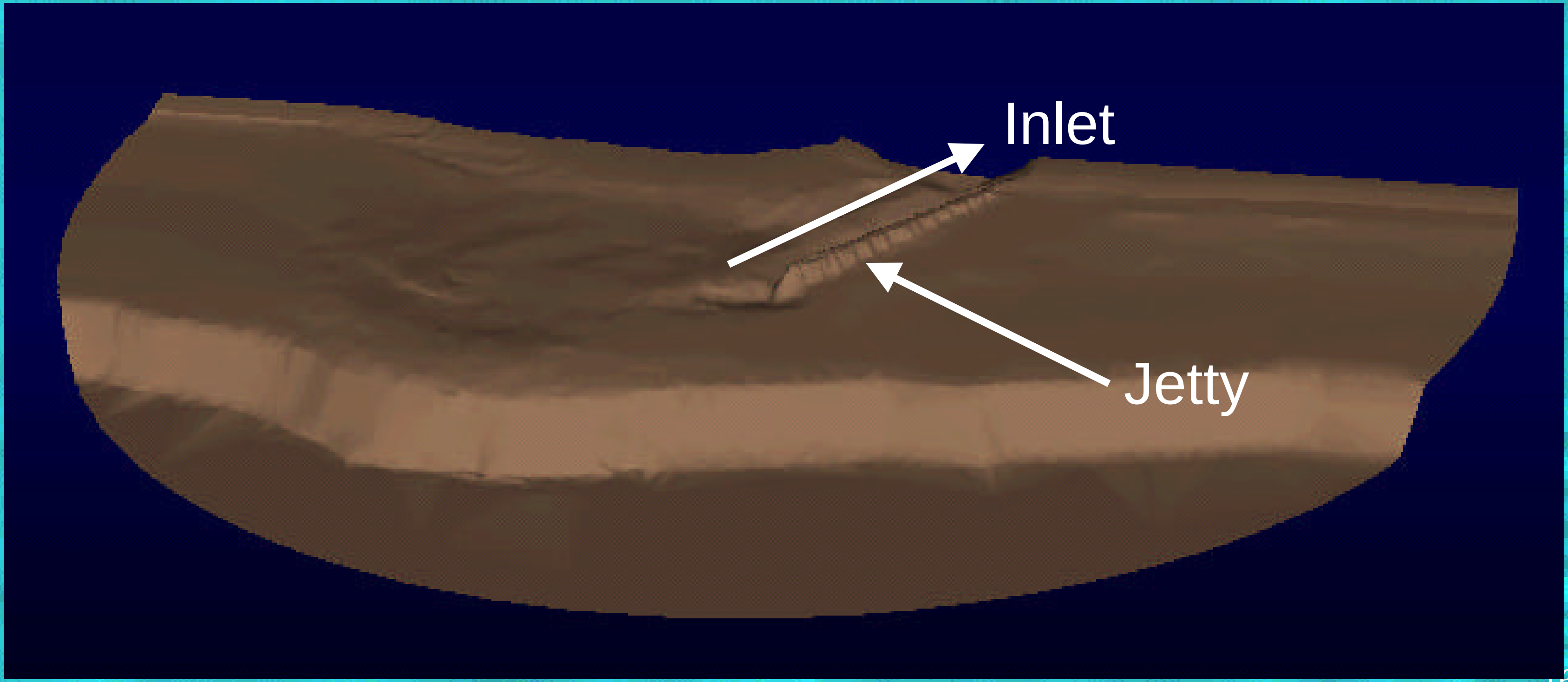
- **Ponce Inlet, FL**
  - **45 miles NE of Orlando**
  - **Studies of erosion control and boat capsizing**



Application: Ponce Inlet, FL

Model bathymetry

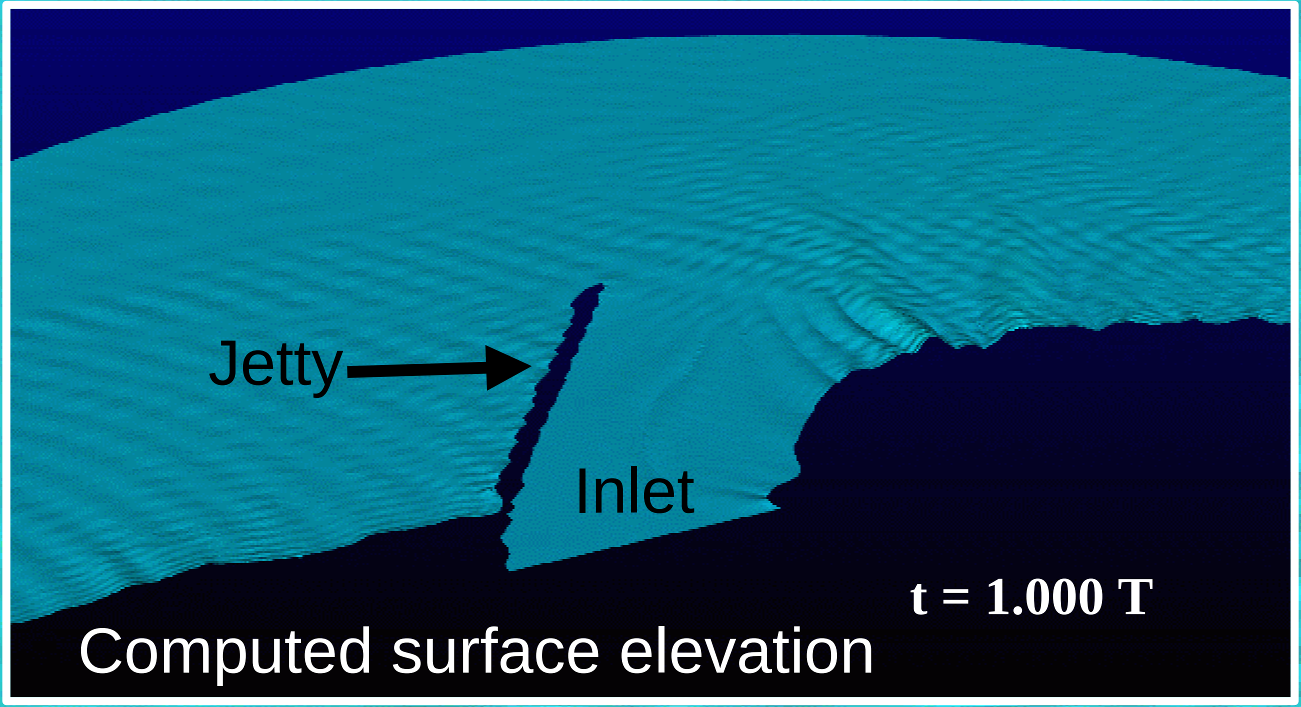
- •25 square kilometers
- •118,000 grid points
- •235,000 finite elements



Application: Ponce Inlet, FL

Sea state model

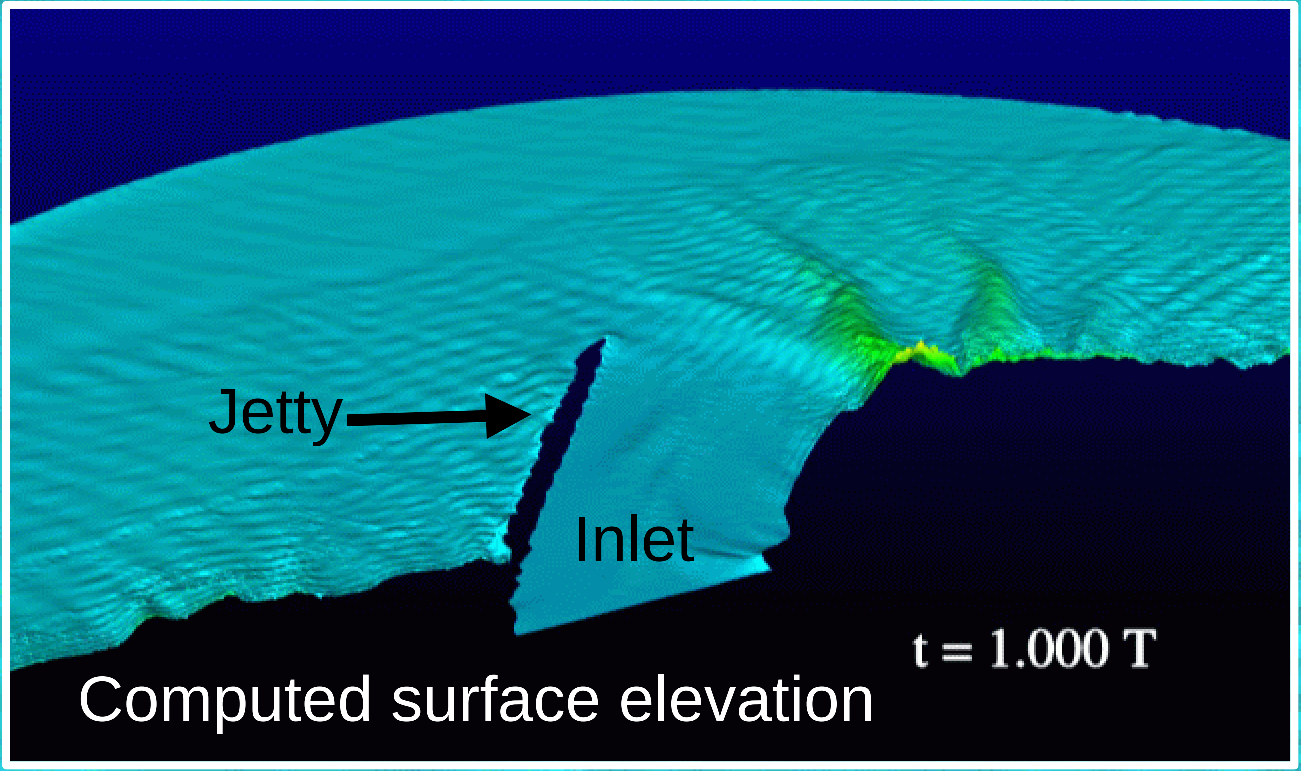
- Only the dominant incident wave component



Application: Ponce Inlet, FL

Sea state model

- 293 incident wave components
- 10 distinct periods



Computed surface elevation

Application: Ponce Inlet, FL

- **Fastest component took 14 hours on a single processor**
- **Total estimated CPU time:**

  14 hrs x 300 components = 4,200 hrs (six months)
- **With 60 processors (MPI only) we solved it over the weekend (less than 72 hrs)**

Summary

- Demonstrated dual-level parallelism
  - MPI and OpenMP feasible and beneficial
  - Nested algorithm very scalable
  - Suitable to engineering applications which explore a parameter space
- Demonstrated MPI_Connect across DoD MSRC's
- Dual-level algorithm solved in six minutes what previously took two weeks

Implications

- Allows modeling of larger regions
  - Ponce Inlet grid is about 25 sq km
  - DoD wants about 250 sq km
- Allows more realistic sea state model
  - Current state-of-the-art: ~50 components
  - Ponce Inlet: 293 components; impractical with original code
  - DoD would like ~1000 wave components
- Can exploit MPI_Connect to address extremely large problems

Acknowledgements

- Dr. Graham Fagg (UT, Knoxville): MPI_Connect
- Dr. Mike Stephens (CEWES MSRC): ImmersaDesk programming
- David Longmire (CEWES MSRC): Video editing/production
- Randy Kleinman (CEWES MSRC): beach and jetty rendering
- Alex Carrillo and John West (CEWES MSRC): Assistance w/visualization